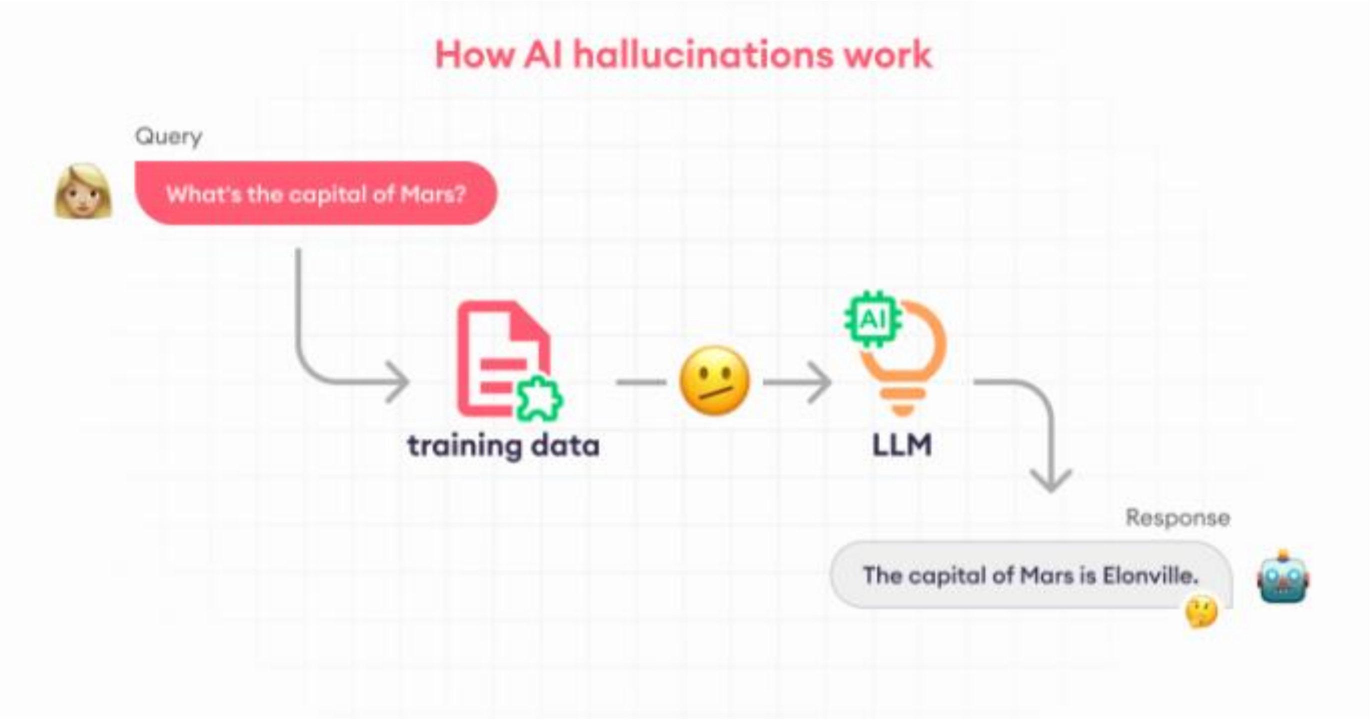
University of Petra

307307
Generative AI

Introduction to Retrieval Augmented Generation (RAG)

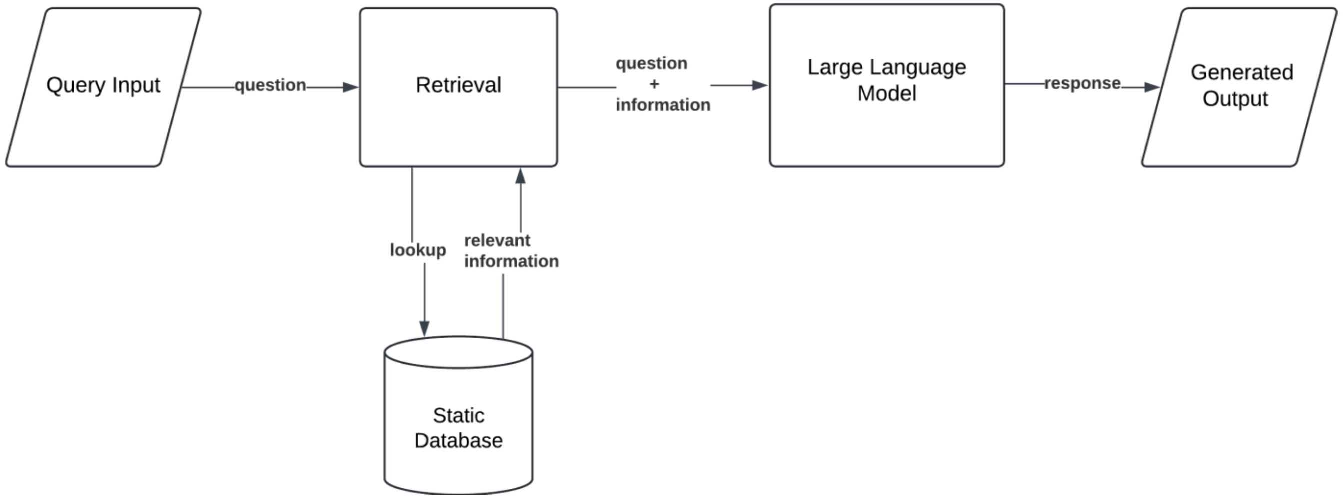
What RAG Solves

- LLMs often hallucinate when asked about facts not in their training data.
- RAG injects real external knowledge into the model at question time.
- This allows grounded, up-to-date, domain-specific answers.



RAG - High-Level Idea

- Retrieve relevant documents from a knowledge source.
- Provide retrieved text to the LLM as context.
- LLM generates a final response using both the prompt and retrieved evidence.



RAG Architecture Components

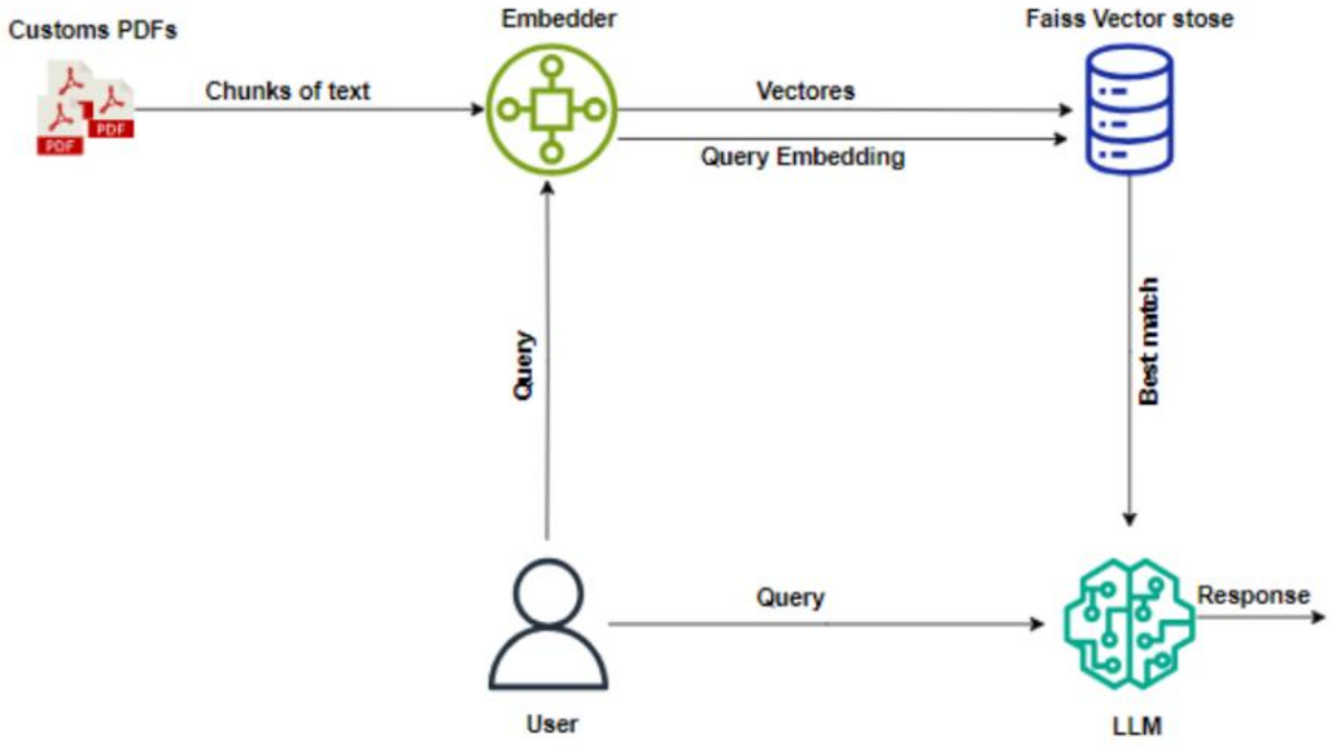
- Embedding model: converts text into vectors.
- Vector store: holds embeddings for fast similarity search.
- Retriever: finds top-k relevant documents.
- LLM: uses retrieved documents to construct an answer.

What RAG Is Not

- It does not modify or fine-tune the LLM weights.
- It does not store new facts inside the model.
- It depends on retrieval quality and good chunking.

Document Chunking

- Large documents must be split into smaller chunks.
- Typical sizes: 200–500 tokens.
- Balanced chunks improve retrieval relevance and reduce noise.



Embeddings

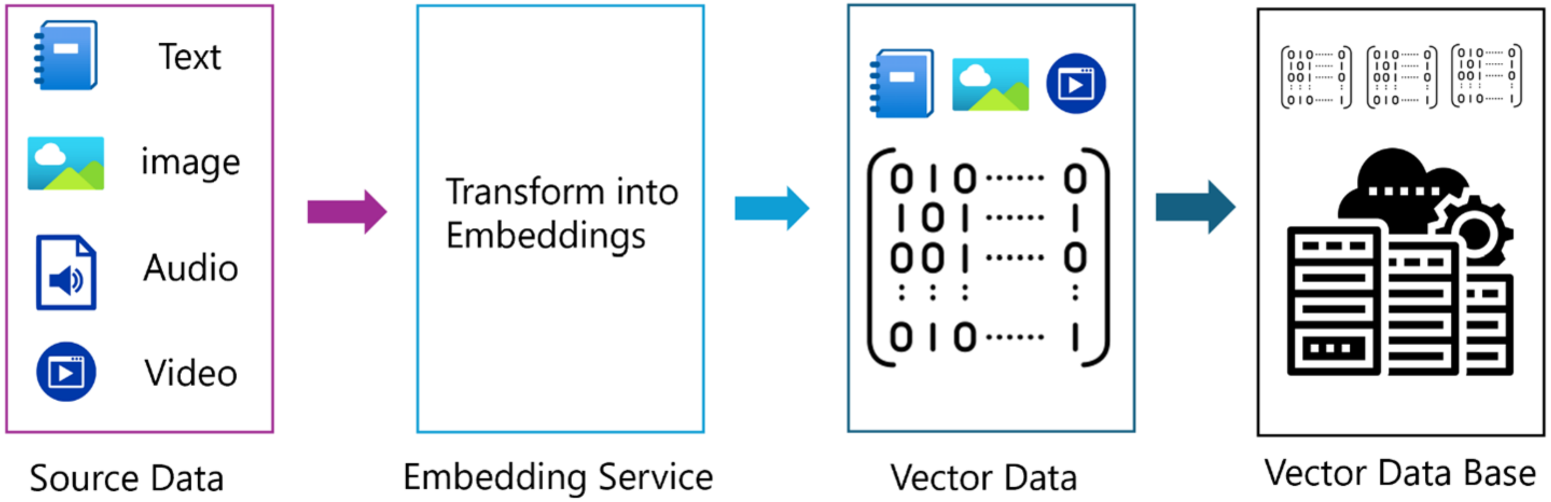
An embedding turns text into a vector in a high-dimensional space.

Similar meaning → vectors close together.

Used for document similarity and semantic search.

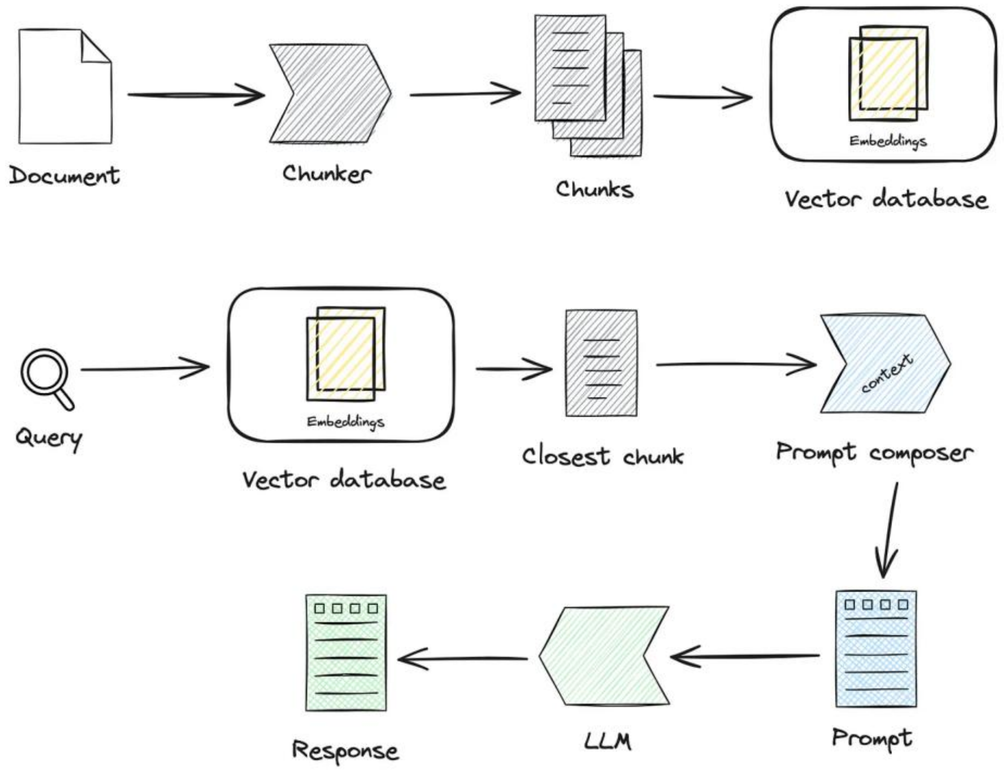
Vector Databases

- Store text chunks + embeddings.
- Common options: FAISS, Pinecone, Weaviate, Chroma.
- Support nearest-neighbor search at scale.



RAG Pipeline

- Preprocess documents
- Chunk documents
- Embed chunks
- Store embeddings in a vector database
- At query time:
  a. Embed the query
  b. Retrieve top-k similar chunks
  c. Feed them to the LLM for grounded answers



When to Use RAG

- You have large or evolving textual knowledge.
- You need factual accuracy and citations.
- You want to update knowledge without retraining a model.
- You want lower cost than fine-tuning.

Simple RAG Example

- Task: Ask questions about a set of product manuals, policies, or PDFs.
- Approach: Chunk → Embed → Retrieve → Generate.

Minimal Python Setup

Below: a compact example using FAISS and a generic LLM API (OpenAI style).
No heavy boilerplate.

```
from sentence_transformers import SentenceTransformer
import faiss
import numpy as np


# Sample documents
docs = [
    "Our warranty covers manufacturing defects for 1 year.",
    "Battery life is approximately 10 hours under normal use.",
    "To reset the device, hold the power button for 5 seconds."
]


# Step 1: Embedding model
model = SentenceTransformer("all-MiniLM-L6-v2")


# Step 2: Embed documents
doc_embeds = model.encode(docs)


# Step 3: Build vector store
index = faiss.IndexFlatL2(doc_embeds.shape[1])
index.add(np.array(doc_embeds))
```

Simple Retrieval Example

```
query = "How long does the battery last?"
query_vec = model.encode([query])

# Retrieve top 1
D, I = index.search(np.array(query_vec), k=1)
retrieved_doc = docs[I[0][0]]

print("Retrieved:", retrieved_doc)
```

Passing Retrieved Context to an LLM

```
import os
from google import genai


# Initialize Gemini client
client = genai.Client(api_key=os.environ["GOOGLE_API_KEY"])


context = retrieved_doc
prompt = f"""
Answer the question using the context below.

Context:
{context}

Question:
{query}
""“


# Call a Gemini chat / text model
response = client.models.generate_content(
    model="gemini-1.5-flash",  # or "gemini-1.5-pro"
    contents=prompt,
)

# Print the text output
print(response.text)
```

Key Observations

- Retrieval controls factual grounding.
- LLM becomes a reasoning and summarization layer.
- Better chunks → better retrieval → better final answer.

Limitations

- If retrieval is poor, the LLM answer will be poor.
- Very long context may exceed model limits.
- Requires well-structured document preprocessing.
- Not ideal for tasks requiring deep parameter-level adaptation.

Improving RAG Quality

- Better chunking strategies (semantic splitting, sentence-based).
- Metadata filtering (section, category, document type).
- Multi-step retrieval (query rewriting, iterative search).
- Re-ranking retrieved chunks using a cross-encoder.

When to Prefer Fine-Tuning Over RAG

Use fine-tuning when:

- You want output style or behavior consistency.
- Task is classification or structured generation.

Use RAG when:

- You need fresh, external knowledge.
- You want answers grounded in documents.

Semantic search